In [2]:

```
# Doanload The Kaggle Data

#!wget --header="Host: storage.googleapis.com" --header="User-Agent: Mozilla/5.0
```

In [ ]:

```
# Refrence Notebook

#https://www.kaggle.com/qcw171717/naive-baseline/
```

In [4]:

```
#!unzip m5-forecasting-accuracy.zip
```

In [5]:

```
# import sum Libaries

import numpy as np
import pandas as pd
from tqdm import tqdm
```

In [85]:

```
df = pd.read_csv('sales_train_evaluation.csv')
df.head()
```

Out[85]:

	id	item_id	dept_id	cat_id	store_id	state_id	d_1
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0

5 rows × 1947 columns

In [86]:

```
IDS = df['id']
```

In [7]:

```
df.shape
```

Out[7]:

```
(30490, 1947)
```

In [8]:

```
price_df = pd.read_csv("sell_prices.csv")
price_df.head()
```

Out[8]:

	store_id	item_id	wm_yr_wk	sell_price
0	CA_1	HOBBIES_1_001	11325	9.58
1	CA_1	HOBBIES_1_001	11326	9.58
2	CA_1	HOBBIES_1_001	11327	8.26
3	CA_1	HOBBIES_1_001	11328	8.26
4	CA_1	HOBBIES_1_001	11329	8.26

In [9]:
```
price_df.shape
```

Out[9]:
```
(6841121, 4)
```

In [10]:
```
cal_df = pd.read_csv("calendar.csv")
cal_df.head()
```

Out[10]:

	date	wm_yr_wk	weekday	wday	month	year	d	event_name_1	event_type_1	event_nar
0	2011-01-29	11101	Saturday	1	1	2011	d_1	NaN	NaN	
1	2011-01-30	11101	Sunday	2	1	2011	d_2	NaN	NaN	
2	2011-01-31	11101	Monday	3	1	2011	d_3	NaN	NaN	
3	2011-02-01	11101	Tuesday	4	2	2011	d_4	NaN	NaN	
4	2011-02-02	11101	Wednesday	5	2	2011	d_5	NaN	NaN	

In [11]:
```
cal_df.shape
```

Out[11]:
```
(1969, 14)
```

In [12]:
```
# Get integer value in d column    ex d_1 , d_2   ---->>>   1  ,1

cal_df["d"]=cal_df["d"].apply(lambda x: int(x.split("_")[1]))
cal_df.head()
```

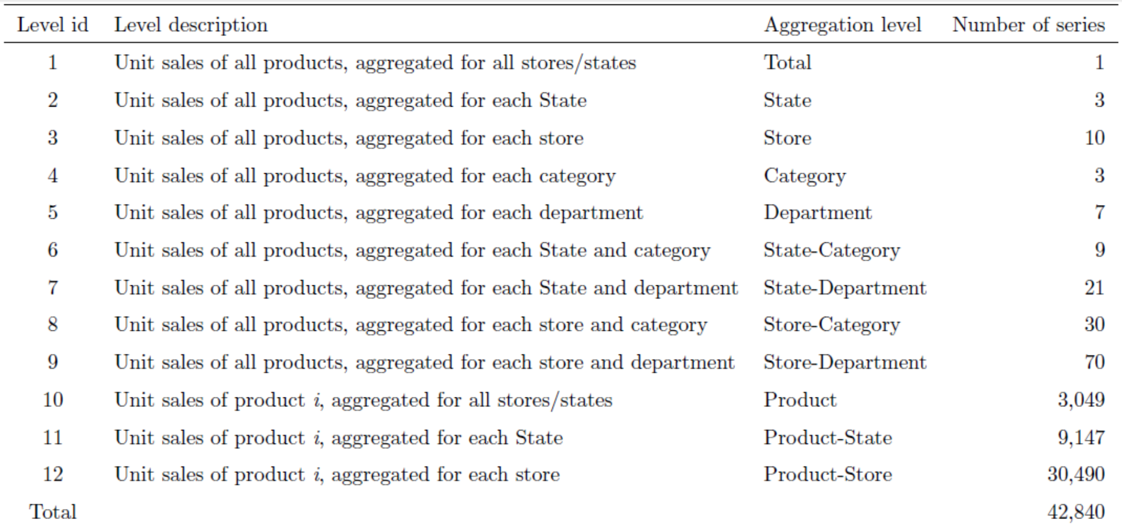
Out[12]:

	date	wm_yr_wk	weekday	wday	month	year	d	event_name_1	event_type_1	event_name
0	2011-01-29	11101	Saturday	1	1	2011	1	NaN	NaN	Na
1	2011-01-30	11101	Sunday	2	1	2011	2	NaN	NaN	Na
2	2011-01-31	11101	Monday	3	1	2011	3	NaN	NaN	Na
3	2011-02-01	11101	Tuesday	4	2	2011	4	NaN	NaN	Na
4	2011-02-02	11101	Wednesday	5	2	2011	5	NaN	NaN	Na

In [13]:
```
price_df["id"] = price_df["item_id"] + "_" + price_df["store_id"] + "_evaluation
price_df.head()
```

Out[13]:

	store_id	item_id	wm_yr_wk	sell_price	id
0	CA_1	HOBBIES_1_001	11325	9.58	HOBBIES_1_001_CA_1_evaluation
1	CA_1	HOBBIES_1_001	11326	9.58	HOBBIES_1_001_CA_1_evaluation
2	CA_1	HOBBIES_1_001	11327	8.26	HOBBIES_1_001_CA_1_evaluation
3	CA_1	HOBBIES_1_001	11328	8.26	HOBBIES_1_001_CA_1_evaluation
4	CA_1	HOBBIES_1_001	11329	8.26	HOBBIES_1_001_CA_1_evaluation



Level id	Level description	Aggregation level	Number of series
1	Unit sales of all products, aggregated for all stores/states	Total	1
2	Unit sales of all products, aggregated for each State	State	3
3	Unit sales of all products, aggregated for each store	Store	10
4	Unit sales of all products, aggregated for each category	Category	3
5	Unit sales of all products, aggregated for each department	Department	7
6	Unit sales of all products, aggregated for each State and category	State-Category	9
7	Unit sales of all products, aggregated for each State and department	State-Department	21
8	Unit sales of all products, aggregated for each store and category	Store-Category	30
9	Unit sales of all products, aggregated for each store and department	Store-Department	70
10	Unit sales of product *i*, aggregated for all stores/states	Product	3,049
11	Unit sales of product *i*, aggregated for each State	Product-State	9,147
12	Unit sales of product *i*, aggregated for each store	Product-Store	30,490
Total			42,840

1. Calculate Weight For Product-Store Level {Level- 12}

In [14]:
```
for day in tqdm(range(1886, 1914)):
    # Get the Week Id of Particular Day
    wk_id = list(cal_df[cal_df["d"]==day]["wm_yr_wk"])[0]
    # Get All Price Information on that Paricular Day
    wk_price_df = price_df[price_df["wm_yr_wk"]==wk_id]
    # Merge Sell Price With Transaction data
    df = df.merge(wk_price_df[["sell_price", "id"]], on=["id"], how='inner')
    # Sales Revnue = Number of Product Sold * Product Price
    df["Sales_Revenue" + str(day)] = df["sell_price"] * df["d_" + str(day)]
    df.drop(columns=["sell_price"], inplace=True)
```

```
100%|██████████| 28/28 [00:08<00:00,  3.36it/s]
```

In [15]:
```
df.head()
```

Out[15]:

	id	item_id	dept_id	cat_id	store_id	state_id	d_1
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0

5 rows × 1975 columns

In [16]:
```
# Get 28 Days Total Revnue by Particular Product

df["dollar_sales"] = df[[c for c in df.columns if c.find("Sales_Revenue")==0]].su
df.head()
```

Out[16]:

	id	item_id	dept_id	cat_id	store_id	state_id	d_1
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0

5 rows × 1976 columns

In [17]:
```
# Drop all the Revenues Columns

df.drop(columns=[c for c in df.columns if c.find("unit_sales")==0], inplace=True
```

In [18]:

```
# Product Contribution in the Revenue

df["weight"] = df["dollar_sales"] / df["dollar_sales"].sum()
df.head()
```

Out[18]:

	id	item_id	dept_id	cat_id	store_id	state_id	d_1
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0

5 rows × 1977 columns

In [19]:

```
df.drop(columns=["dollar_sales"], inplace=True)
```

In [20]:

```
df["weight"] /= 12
```

2. Forecasting Next 28 days Using Simple Moving Average

In [21]:

```
all_days_col = [h for h in df.columns if 'd_' in h]
print("First 5 values ",all_days_col[0:5])
print("Last 5 values ",all_days_col[-5:])
```

```
First 5 values  ['d_1', 'd_2', 'd_3', 'd_4', 'd_5']
Last 5 values  ['d_1937', 'd_1938', 'd_1939', 'd_1940', 'd_1941']
```

In [22]:

```
train_data = df[all_days_col[:1913]]
train_data.head()
```

Out[22]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	d_11	d_12	d_13	d_14	d_15	d_16	d_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 1913 columns

In [23]:
```
val_data = df[all_days_col[1913:]]
val_data.head()
```

Out[23]:

	d_1914	d_1915	d_1916	d_1917	d_1918	d_1919	d_1920	d_1921	d_1922	d_1923	d_1924	d_
0	0	0	0	2	0	3	5	0	0	1	1	
1	0	1	0	0	0	0	0	0	0	1	0	
2	0	0	1	1	0	2	1	0	0	0	0	
3	0	0	1	2	4	1	6	4	0	0	0	
4	1	0	2	3	1	0	3	2	3	1	1	

In [24]:
```
def simple_Moving_Average(train_data,  forecast_days,window_Size):

    predictions = []
    for i in range(forecast_days):
        # All Data Avilabel in Trainin Data
        if i == 0:
            predictions.append(np.mean(train_data[train_data.columns[-window_Siz
        if i < forecast_days and i > 0:
            predictions.append((np.sum(train_data[train_data.columns[-window_Siz
                                      np.sum(predictions[:i], axis=0))/forecast_days

    return predictions
```

In [25]:
```
forecast_days = 28
window_Size = 28
predictions = simple_Moving_Average(train_data, forecast_days, window_Size)
```

In [26]:
```
for d, i in enumerate(range(1914, 1942)):
    df['F_' + str(i)] = predictions[d]
```

In [27]:
```
df.head()
```

Out[27]:

	id	item_id	dept_id	cat_id	store_id	state_id	d_1
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0

5 rows × 2004 columns

In [ ]:
```
# --->> Level 12 Ground Truth Vaulues And Forecasting Values we have.
```

3. Focus on Higher Level Aggregating

Level 1. Aggregation of Total

In [28]:
```
data = df[[a for a in df.columns if a.find("d_") == 0 or a.find("F_") == 0]]
# Get All Columns Sum
data = data.sum()
# Transpose the data
aggregated_df = pd.DataFrame(data).transpose()
aggregated_df
```

Out[28]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	d_1
0	32631.0	31749.0	23783.0	25412.0	19146.0	29211.0	28010.0	37932.0	32736.0	25572.0	23071.

1 rows × 1969 columns

In [29]:
```
aggregated_df["level"] = 1
aggregated_df["weight"] = 1/12
aggregated_df
```

Out[29]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	d_1
0	32631.0	31749.0	23783.0	25412.0	19146.0	29211.0	28010.0	37932.0	32736.0	25572.0	23071.

1 rows × 1971 columns

In [ ]:
```
# we will Give Each Level Around 8.3% Weight
```

Level-2 To Level-11 Aggregation

In [30]:
```
aggregation_level = {2: ["state_id"],
                     3: ["store_id"],
                     4: ["cat_id"],
                     5: ["dept_id"],
                     6: ["state_id", "cat_id"],
                     7: ["state_id", "dept_id"],
                     8: ["store_id", "cat_id"],
                     9: ["store_id", "dept_id"],
                     10: ["item_id"],
                     11: ["item_id", "state_id"]}
```

In [31]:

```
columns = aggregated_df.columns

for lev in aggregation_level:
    # Group by Based on Aggregation Level
    new_df = df.groupby(by=aggregation_level[lev]).sum().reset_index()
    # Add Level Column
    new_df["level"] = lev
    # Append your new DataFrame into old DataFrame
    aggregated_df = aggregated_df.append(new_df[columns])
```

In [32]:

```
aggregated_df
```

Out[32]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	
0	32631.0	31749.0	23783.0	25412.0	19146.0	29211.0	28010.0	37932.0	32736.0	25572.0	23
0	14195.0	13805.0	10108.0	11047.0	9925.0	11322.0	12251.0	16610.0	14696.0	11822.0	10
1	9438.0	9630.0	6778.0	7381.0	5912.0	9006.0	6226.0	9440.0	9376.0	7319.0	6
2	8998.0	8314.0	6897.0	6984.0	3309.0	8883.0	9533.0	11882.0	8664.0	6431.0	5
0	4337.0	4155.0	2816.0	3051.0	2630.0	3276.0	3450.0	5437.0	4340.0	3157.0	2
...	...	...	...	...	...	...	...	...	...	...	
9142	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9143	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	
9144	0.0	2.0	0.0	1.0	1.0	1.0	0.0	2.0	1.0	2.0	
9145	2.0	1.0	0.0	0.0	0.0	0.0	1.0	1.0	0.0	0.0	
9146	0.0	1.0	0.0	2.0	0.0	1.0	1.0	1.0	0.0	0.0	

12350 rows × 1971 columns

In [33]:

```
print(df.shape[0]+aggregated_df.shape[0])
```

```
42840
```

In [ ]:

```
# For Each Level i have my Ground Truth Value And Forecasting value
```

4. Calculalte RMSSE and WRMSSE

$$RMSSE = \sqrt{\frac{\frac{1}{h}\sum_{t=n+1}^{n+h}(y_t - \hat{y}_t)^2}{\frac{1}{n-1}\sum_{t=2}^{n}(y_t - y_{t-1})^2}},$$

```
In [34]: h = 28        # Forecasting For 28 Days
         n = 1913      # my Traning Data
```

```
In [35]: def RMSSE(ground_truth, forecast, train_series):

             num = ((ground_truth - forecast)**2).sum(axis=1)
             den = 1/(n-1) * ((train_series[:, 1:] - train_series[:, :-1]) ** 2).sum(axis
             rmsse = (1/h * num/den) ** 0.5

             return rmsse
```

```
In [36]: # First 1913 Days Columns

         train_series_cols = [c for c in df.columns if c.find("d_") == 0][:-28]
         train_series_cols[-5:]

Out[36]: ['d_1909', 'd_1910', 'd_1911', 'd_1912', 'd_1913']
```

```
In [37]: # 28 Days Columns

         ground_truth_cols = [c for c in df.columns if c.find("d_") == 0][-28:]
         ground_truth_cols[-5:]

Out[37]: ['d_1937', 'd_1938', 'd_1939', 'd_1940', 'd_1941']
```

```
In [38]: # Forecasting Columns

         forecast_cols = [c for c in df.columns if c.find("F_") == 0]
         forecast_cols[-5:]

Out[38]: ['F_1937', 'F_1938', 'F_1939', 'F_1940', 'F_1941']
```

In [39]:

```
# For Level 12 Calculate RMSSE

df["rmsse"] = RMSSE(np.array(df[ground_truth_cols]),
                    np.array(df[forecast_cols]), np.array(df[train_series_cols]))

df.head()
```

Out[39]:

	id	item_id	dept_id	cat_id	store_id	state_id	d_1
0	HOBBIES_1_001_CA_1_evaluation	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0
1	HOBBIES_1_002_CA_1_evaluation	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0
2	HOBBIES_1_003_CA_1_evaluation	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0
3	HOBBIES_1_004_CA_1_evaluation	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0
4	HOBBIES_1_005_CA_1_evaluation	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0

5 rows × 2005 columns

In [40]:

```
# For Level 1 to 11 Calculate RMSSE

aggregated_df["rmsse"] = RMSSE(np.array(aggregated_df[ground_truth_cols]),
                    np.array(aggregated_df[forecast_cols]), np.array(aggregated_d

aggregated_df.head()
```

Out[40]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	d_1
0	32631.0	31749.0	23783.0	25412.0	19146.0	29211.0	28010.0	37932.0	32736.0	25572.0	23071.
0	14195.0	13805.0	10108.0	11047.0	9925.0	11322.0	12251.0	16610.0	14696.0	11822.0	10933.
1	9438.0	9630.0	6778.0	7381.0	5912.0	9006.0	6226.0	9440.0	9376.0	7319.0	6224.
2	8998.0	8314.0	6897.0	6984.0	3309.0	8883.0	9533.0	11882.0	8664.0	6431.0	5914.
0	4337.0	4155.0	2816.0	3051.0	2630.0	3276.0	3450.0	5437.0	4340.0	3157.0	2995.

5 rows × 1972 columns

In [42]:

```
# Calculate WRMSSE

df["wrmsse"] = df["weight"] * df["rmsse"]
aggregated_df["wrmsse"] = aggregated_df["weight"] * aggregated_df["rmsse"]
```

In [43]:
```
df["wrmsse"].sum() + aggregated_df["wrmsse"].sum()
```

Out[43]:
```
1.0970029012597868
```

Prediction Part

In [73]:
```
all_days_col = [h for h in df.columns if 'd_' in h]
print("First 5 values ",all_days_col[0:5])
print("Last 5 values ",all_days_col[-5:])

df = df[all_days_col]
df.head()
```

```
First 5 values  ['d_1', 'd_2', 'd_3', 'd_4', 'd_5']
Last 5 values  ['d_1965', 'd_1966', 'd_1967', 'd_1968', 'd_1969']
```

Out[73]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	d_11	d_12	d_13	d_14	d_15	d_16	d_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 1969 columns

In [74]:
```
# Add Test Data

for day in range(1942,1970):
  df['d_' + str(day)] = 0
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:4: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
  after removing the cwd from sys.path.
```

In [75]:
```
df.head()
```

Out[75]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	d_11	d_12	d_13	d_14	d_15	d_16	d_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 1969 columns

In [76]:
```
all_days_col = [h for h in df.columns if 'd_' in h]
print("First 5 values ",all_days_col[0:5])
print("Last 5 values ",all_days_col[-5:])

df = df[all_days_col]
df.head()
```

```
First 5 values  ['d_1', 'd_2', 'd_3', 'd_4', 'd_5']
Last 5 values  ['d_1965', 'd_1966', 'd_1967', 'd_1968', 'd_1969']
```

Out[76]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	d_11	d_12	d_13	d_14	d_15	d_16	d_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 1969 columns

In [77]:
```
train_data = df[all_days_col[:1941]]
train_data.head()
```

Out[77]:

	d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	d_11	d_12	d_13	d_14	d_15	d_16	d_1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

5 rows × 1941 columns

In [78]:

```
val_data = df[all_days_col[1941:]]
val_data.head()
```

Out[78]:

	d_1942	d_1943	d_1944	d_1945	d_1946	d_1947	d_1948	d_1949	d_1950	d_1951	d_1952	d_
0	0	0	0	0	0	0	0	0	0	0	0	
1	0	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	0	0	
4	0	0	0	0	0	0	0	0	0	0	0	

In [79]:

```
def simple_Moving_Average(train_data,  forecast_days,window_Size):

    predictions = []
    for i in range(forecast_days):
        # All Data Avilabel in Trainin Data
        if i == 0:
            predictions.append(np.mean(train_data[train_data.columns[-window_Size
        if i < forecast_days and i > 0:
            predictions.append((np.sum(train_data[train_data.columns[-window_Size
                                np.sum(predictions[:i], axis=0))/forecast_days

    return predictions
```

In [80]:

```
forecast_days = 28
window_Size = 28
predictions = simple_Moving_Average(train_data, forecast_days, window_Size)
```

In [81]:

```
for d, i in enumerate(range(1942, 1970)):
    val_data['d_' + str(i)] = predictions[d]
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:2: SettingWithCopy
Warning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pyd
ata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
opy)
```

In [82]:
```
val_data
```

Out[82]:

	d_1942	d_1943	d_1944	d_1945	d_1946	d_1947	d_1948	d_1949	d_1950
0	1.178571	1.220663	1.264258	1.309410	1.284747	1.330630	1.271010	1.137832	1.178469
1	0.250000	0.258929	0.232462	0.240764	0.249363	0.258268	0.267492	0.277046	0.286940
2	0.750000	0.776786	0.804528	0.797547	0.790316	0.818542	0.776347	0.768360	0.795801
3	1.750000	1.812500	1.877232	1.908562	1.905296	1.830485	1.860146	1.712294	1.630590
4	1.392857	1.406888	1.457134	1.437746	1.381951	1.395592	1.445435	1.389914	1.368126
...	...	...	...	...	...	...	...	...	...
30485	0.642857	0.665816	0.689595	0.714224	0.668303	0.620743	0.642912	0.665873	0.689654
30486	0.285714	0.295918	0.270773	0.244729	0.217755	0.225532	0.233586	0.241929	0.250569
30487	0.785714	0.813776	0.842839	0.837226	0.831413	0.861106	0.820431	0.814018	0.807376
30488	1.321429	1.332908	1.273369	1.318847	1.330234	1.306314	1.317254	1.364298	1.341595
30489	1.250000	1.294643	1.340880	1.388769	1.438368	1.489738	1.507228	1.525344	1.544106

30490 rows × 28 columns

In [91]:
```
p1 = pd.concat([IDS, val_data], axis=1, sort=False)
p1
```

Out[91]:

	id	d_1942	d_1943	d_1944	d_1945	d_1946	d_1947
0	HOBBIES_1_001_CA_1_evaluation	1.178571	1.220663	1.264258	1.309410	1.284747	1.33063
1	HOBBIES_1_002_CA_1_evaluation	0.250000	0.258929	0.232462	0.240764	0.249363	0.25826
2	HOBBIES_1_003_CA_1_evaluation	0.750000	0.776786	0.804528	0.797547	0.790316	0.81854
3	HOBBIES_1_004_CA_1_evaluation	1.750000	1.812500	1.877232	1.908562	1.905296	1.83048
4	HOBBIES_1_005_CA_1_evaluation	1.392857	1.406888	1.457134	1.437746	1.381951	1.39559
...	...	...	...	...	...	...	..
30485	FOODS_3_823_WI_3_evaluation	0.642857	0.665816	0.689595	0.714224	0.668303	0.62074
30486	FOODS_3_824_WI_3_evaluation	0.285714	0.295918	0.270773	0.244729	0.217755	0.22553
30487	FOODS_3_825_WI_3_evaluation	0.785714	0.813776	0.842839	0.837226	0.831413	0.86110
30488	FOODS_3_826_WI_3_evaluation	1.321429	1.332908	1.273369	1.318847	1.330234	1.30631
30489	FOODS_3_827_WI_3_evaluation	1.250000	1.294643	1.340880	1.388769	1.438368	1.48973

30490 rows × 29 columns

In [92]:
```
p2 = p1.copy()

p2["id"] = p1["id"].str.replace("evaluation$", "validation")
p2
```

Out[92]:

	id	d_1942	d_1943	d_1944	d_1945	d_1946	d_1947
0	HOBBIES_1_001_CA_1_validation	1.178571	1.220663	1.264258	1.309410	1.284747	1.330630
1	HOBBIES_1_002_CA_1_validation	0.250000	0.258929	0.232462	0.240764	0.249363	0.258268
2	HOBBIES_1_003_CA_1_validation	0.750000	0.776786	0.804528	0.797547	0.790316	0.818542
3	HOBBIES_1_004_CA_1_validation	1.750000	1.812500	1.877232	1.908562	1.905296	1.830485
4	HOBBIES_1_005_CA_1_validation	1.392857	1.406888	1.457134	1.437746	1.381951	1.395592
...	...	...	...	...	...	...	...
30485	FOODS_3_823_WI_3_validation	0.642857	0.665816	0.689595	0.714224	0.668303	0.620743
30486	FOODS_3_824_WI_3_validation	0.285714	0.295918	0.270773	0.244729	0.217755	0.225532
30487	FOODS_3_825_WI_3_validation	0.785714	0.813776	0.842839	0.837226	0.831413	0.861106
30488	FOODS_3_826_WI_3_validation	1.321429	1.332908	1.273369	1.318847	1.330234	1.306314
30489	FOODS_3_827_WI_3_validation	1.250000	1.294643	1.340880	1.388769	1.438368	1.489738

30490 rows × 29 columns

In [93]:
```
p3 = pd.concat([p1, p2], axis=0, sort=False)
p3
```

Out[93]:

	id	d_1942	d_1943	d_1944	d_1945	d_1946	d_1947
0	HOBBIES_1_001_CA_1_evaluation	1.178571	1.220663	1.264258	1.309410	1.284747	1.33063
1	HOBBIES_1_002_CA_1_evaluation	0.250000	0.258929	0.232462	0.240764	0.249363	0.25826
2	HOBBIES_1_003_CA_1_evaluation	0.750000	0.776786	0.804528	0.797547	0.790316	0.81854
3	HOBBIES_1_004_CA_1_evaluation	1.750000	1.812500	1.877232	1.908562	1.905296	1.83048
4	HOBBIES_1_005_CA_1_evaluation	1.392857	1.406888	1.457134	1.437746	1.381951	1.39559
...	...	...	...	...	...	...	..
30485	FOODS_3_823_WI_3_validation	0.642857	0.665816	0.689595	0.714224	0.668303	0.62074
30486	FOODS_3_824_WI_3_validation	0.285714	0.295918	0.270773	0.244729	0.217755	0.22553
30487	FOODS_3_825_WI_3_validation	0.785714	0.813776	0.842839	0.837226	0.831413	0.86110
30488	FOODS_3_826_WI_3_validation	1.321429	1.332908	1.273369	1.318847	1.330234	1.30631
30489	FOODS_3_827_WI_3_validation	1.250000	1.294643	1.340880	1.388769	1.438368	1.48973

60980 rows × 29 columns

In [94]:
```
p3.columns = ['id'] + ['F' + str(c) for c in np.arange(1,29,1)]
```

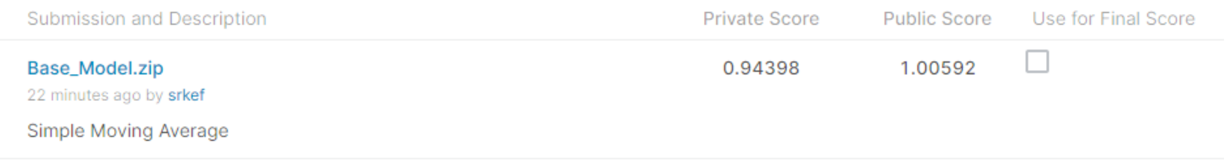
In [95]:
```
p3
```

Out[95]:

	id	F1	F2	F3	F4	F5	F6
0	HOBBIES_1_001_CA_1_evaluation	1.178571	1.220663	1.264258	1.309410	1.284747	1.330630
1	HOBBIES_1_002_CA_1_evaluation	0.250000	0.258929	0.232462	0.240764	0.249363	0.258268
2	HOBBIES_1_003_CA_1_evaluation	0.750000	0.776786	0.804528	0.797547	0.790316	0.818542
3	HOBBIES_1_004_CA_1_evaluation	1.750000	1.812500	1.877232	1.908562	1.905296	1.830485
4	HOBBIES_1_005_CA_1_evaluation	1.392857	1.406888	1.457134	1.437746	1.381951	1.395592
...	...	...	...	...	...	...	..
30485	FOODS_3_823_WI_3_validation	0.642857	0.665816	0.689595	0.714224	0.668303	0.620743
30486	FOODS_3_824_WI_3_validation	0.285714	0.295918	0.270773	0.244729	0.217755	0.225532
30487	FOODS_3_825_WI_3_validation	0.785714	0.813776	0.842839	0.837226	0.831413	0.861106
30488	FOODS_3_826_WI_3_validation	1.321429	1.332908	1.273369	1.318847	1.330234	1.306314
30489	FOODS_3_827_WI_3_validation	1.250000	1.294643	1.340880	1.388769	1.438368	1.489738

60980 rows × 29 columns

In [96]:
```
p3.to_csv("Base_Model.csv",index=False)
```



Submission and Description	Private Score	Public Score	Use for Final Score
Base_Model.zip 22 minutes ago by srkef Simple Moving Average	0.94398	1.00592	□

In [ ]:
```
# This Is Base Model,Less Score Then This is not Acceptable.
```